MODIS QUARTERLY REPORT
- MARCH 1995-

UNIVERSITY OF MIAMI
RSMAS/MPO

DR. ROBERT H. EVANS

NAS5-31362

==========

Due to the interlocking nature of a number of projects, this and subsequent reports will contain coding to reflect the funding source. Modis funded activities are designated with an **M**, SeaWIFS with an **S**, and Pathfinder with a **P**. There are several major sections within this report; Database, client/server, matchup database, and DSP support.

A. NEAR TERM OBJECTIVES
B. OVERVIEW OF CURRENT PROGRESS
C. FUTURE ACTIVITIES
D. PROBLEMS

A. NEAR TERM OBJECTIVES

A.1 Modis Objectives (M)

A.1.1. Continue to develop and expand the processing environment

a. increase computational efficiency through concurrent operations

b. determine and apply more efficient methods of data availability for processes

A.1.2. Begin extensive testing using global CZCS and AVHRR GAC data with database processing to test the following:

a. algorithm capability

b. machine and operating system stability

c. functionality required for the processing and analysis environment

A.2 SeaWIFS Objectives (S)

A.2.1. Continue testing of processing methodology.

A.2.2. Continue to develop relationship between database and *in-situ* environment.

A.3 Pathfinder Objectives (P)

A.3.1. Expand matchup database as applicable.

A.3.2. Continue testing of methodology.

A.3.3 Train and integrate new personnel into Matchup Database processing scheme.

A.4 DSP Objectives (M)

A.4.1. Continue testing of processing methodology.

A.4.2. Continue to expand the number of sites supported.

A.4.3. Expand the supported hardware/software platforms

B. OVERVIEW OF CURRENT PROGRESS

B.1 Automatic Processing Database (P)

B.1.1 Operational Testing

B1.1.1 January Operational Testing

Data for 1991 was now available on our new digital tape library machine, so this was processed. A large number of new pieces of equipment were added to the processing environment, and the processing slowed while these were integrated into the processing.

A new 4-processor DEC 2100 computer was tested as the primary processor, but numerous problems were encountered. Near the end of the month, the processing was returned to the single-processor alphas.

B.1.1.2 February Operational Testing

Processing continued on the 1991 time period, but was slow due to sever disk and interface problems.

B.1.1.3 March Operational Testing

Processing continued in the 91 time range, covering 91160-91250, using the yearly coefficients for the atmospheric correction. This covered, for the first time, the post-Pinetubo time period. This processing stream was terminated when it became clear that the yearly coefficients were not producing acceptable results for this period.

Processing was discontinued on modis, the 4-processor Sable, when it became clear that disk controller problems were severe, and would not allow continuous use during the processing. Processing was returned to four single-processor Alphas.

Two more single-processor alphas were added to the stable of machines processing orbits, totaling six. Subsequently, the pace of the processing increased.

The time period 91001-91166 was recalculated, using the new technique of monthly coefficients for the atmospheric correction step.

B.1.2 Development

B.1.2.1 January Development.

There were numerous changes, both small and large, to accommodate the new equipment, which included a 4-processor DEC 2100 computer, a digital tape library (used both for spooling input data and backup of output files), and a large amount of new disks.

However, the increase in capabilities also caused a commensurate increase in system problems. In particular, the problems previously encountered with "drop out" of NFS-mounted disks became untenable when modis, the 4-processor Sable, was used for processing. The most damaging point was the *.sh and *.dsp files that are written by the VMS APServer and read by the UNIX processor. While there had previously been occasional dropout at this point, this step was failing 3-5% of the time. A new technique was implemented, whereby the APServer writes the *.sh/*.dsp files to a local disk, rshells a job to the processing computer, which

transfers the command files to one of its local disks. This method did overcome the problems at this step.

Since the input data files will now be supplied by the DLT device (digital tape library) on UNIX, a number of new procedures and command files were developed to spool the data off, and copy it to the VMS side.

As we develop experience with the new paradigm, more changes will be needed. For example, modifications will be needed to eliminate the need to copy the input file to the VMS disks. Currently, the GETSCAN program extracts the scan lines of the pole crossings, and stores this information in a flat file containing information for a whole year for a given satellite. This scan line information is then extracted from the flat file by the ADDREC program, which decides the pieces to be processed, using the asc/dsc pole crossings, and a set of rules for piece size. This information is used to add the processing records to the database.

We are developing a version of GETSCAN that combines elements of both the current GETSCAN and ADDREC. It uses the pole crossing information to find the scan lines of the pole crossings, breaks the pass up into a few pieces, and writes an ingester input file for each piece.

To fuse the two systems, changes will need to be made to GETSCAN and to ADDREC. Much of the functionality, exclusive of the actual record addition, must be moved to GETSCAN, to eliminate the need for a copy of the input file. These include

Extracting the exact ingester name corresponding to a scan line.
Incorporating the full set of rules to decides pieces.
Extract any other information from the datafile that is needed.

While the list is small, the information is not so easily defined. We will need to pick through ADDREC a lot to make sure we are getting all that we need, and only what we need.

Currently, the presence of the input file starts the GETSCAN/ADDREC process. We will have the new GETSCAN put out one ASCII file that contains the information needed for record addition. This file will be rcp'd to the VMS input directory where its appearance will start the

ADDREC process. This file will be used by a new streamlined version of ADDREC to add the processing records to the database.

Summary:

	Current	New
read asc/dsc file	GETSCAN	GETSCAN
find scan lines of pole crossings	GETSCAN	GETSCAN
find filename of that scan line	GETSCAN	GETSCAN
store scn ln/flnm of pole crossings	GETSCAN	
read scan line/filename, pole crossing info	ADDREC	
decide on pieces to be created	ADDREC	GETSCAN
store information on pieces		GETSCAN
read information on pieces		ADDREC
add records to db		ADDREC

These changes will be implemented in the next few months.

B.1.2.2 February Development

Most development work in this time period still related to the integration of the new equipment, and modifying the processing methods to both make best advantage of the new capabilities and to overcome problems.

Processing this month continued on the 4-processor Sable, modis, but many more adjustments were needed.

File transfers using rcp were becoming unreliable, so the code and command files will need to be changed to use ftp for file transfer.

A major revision of the command files and procedures is in progress. These will be covered in the next month's report, when they are complete.

B.1.2.3 March Development.

All file transfer was changed from rcp call to ftp.

There has been considerable consolidation and shortening of commands.

Many of the commands can be issued from any computer - i.e., even checking and control of the server status can be done from UNIX.

There used to be a large number of single-use command files - many of these (but not all) have been consolidated into single command files that interpret input parameters. These changes will be covered in a separate section.

The new system will not seem much different. The major differences OPERATIONALLY are changes in the daily and weekly job triggering, changes in some directories, and changes in how mcp is started.

MCP:

The old need for different mcp binaries for each job type has been eliminated. A single mcp is used, and the 'psa' command has been modified to show the command line in the ps grid, showing which job type is being run.

The mcp-t job, which ran the recipe called GAC_PTB, has been changed to reflect its function - it is now mcp/orbit, and the recipe name is GAC_ORBIT.

Previously, there were separate files to start each job or combination of jobs (starti, startu1, startust, etc.). There is now one 'start'

command file that takes a single parameter, the type of jog or jobs to start. Currently, the choices are:

Single: i, u1, s1, o, d, w

Combined: uso - u1, s1 & o
us - u1 & s1

The start command file can be rsh'ed to the machine you want to start from andrew.

DIRECTORIES

The pass_time and ascdsc files are now in a directory pointed to by ap_etc, instead of mcp_etc. The ap_autoproc_computer_##.sh .dsp and .rpt files, as well as the data transfer files for the record adder now go into dsp_usr2:[ap.tmp.alexis]. (Or in dsp_usr2:[ap.tmp.mariah] for the mariah system).

DAILY/WEEKLY TRIGGERING

Previously, when all passes for a given yearday were done, the daily job for two days before was triggered. This permitted daily jobs to run before all passes for that day were processed, causing stray orbits to pop up at times.

The new triggering is:

Daily:

When each orbit is done, the server checks to see if that day is complete. If not, no triggering occurs. If complete, then:

Three day periods are checked to see if a daily job is ready.

For example, say day n has just been completed:

Check if day n-1 is done. If so, check if day n-2 is done. If n, n-1 and n-2 are done, trigger daily for n-1.

Check if day n+1 is done. If so, check if day n+2 is done. If n, n+1 and n+2 are done, trigger daily for n+1.

Days n-1 and n+1 were checked. If n-1, n and n+1 are all done, trigger daily for day n.

Weekly:

When a daily completes, the server checks to see if all days of that week are done. If true, the weekly1 job is triggered.

When the weekly1 job is done, the server checks to see if the previous and following weeks are done. If so, the weekly2 job is triggered.

Completion of the weekly2 job triggers the weeky3 job, which is allowed to run only on the machine servicing the volume set for that week.

Completion of the weekly3 job triggers the weekly4 job.

VMS Commands

The old way to start the VMS batch jobs has been shortened a bit. The old '@ap_com:submit_apserver' command has been replaced by a simple 'apserver'.

All the new commands to start UNIX are:

```
$ apserver          (was: @ap_com:submit_apserver)
$ addrecgac         (was: @ap_com:submit_addrecgac)
$ getscangac        (was: @ap_com:submit_getscangac)
$ fixscangac        (was: @ap_com:submit_fixscangac)
$ mvgetscan         (was: @ap_com:submit_mvgetscan)
```

Also as before, the MCP control is in the dbrequest_* and run_* logicals, with the DBREQUEST_STATUS controlling the mcps from all processing machines., and the RUN_abrv controlling the mcps from a single machine (ie, RUN_KEL controls KELSO).

However, now a single command file, 'mcp.com', can manipulate both the dbrequest_* and run_* logicals. An input of 'mcp' will list while 'mcp ?' provides a short explanation. To change the global value, input the change as the first input parameter. (To stop mcp's , say 'mcp stop'.) To change the values for only one computer, then use 'mcp option computer'. You can use either the full computer name or

an "accepted" abbreviation. (The "accepted abbreviations are listed in the db loading file ap_dbdef:computer_load.sql. This file can be found AFTER invoking the home:ap_build.com command file, which defines the directories for the db and server.)

For checking the db records, there are still a number of *.sql files that can be used inside sql, or are called by a DCL command file. However, they are all called by a single command file, ck.com, which also has a symbol defined, so that from ANY directory, you simple invoke ck and input a parameter. This ck command file first checks to see if the parameter is one of it's "known" commands. If it is, ck calls the right command file in sql. If the input parameter is not recognized as a command, it is assumed to be the name or abbreviation of a computer, and the jobs assigned to that computer are checked.

The queries that are recognized are:

MAINLINK (can input a main_link, or be prompted)
DAILY
DSPA
EXE
INIT
ORBIT
ORBIN (BOTH ORBIT AND INIT)
SUB
HSUB
UNF
WEEKLY
WEEKLY1 or W1
WEEKLY2 or W2
WEEKLY3 or W3
WEEKLY4 or W4
FOURWK or FW
WEEKLYSUB

Note that the MAINLINK query can take a main_link as the second input parameter, but that ck prompts for it if omitted from the command line.

VMS/UNIX CROSSOVER

Both mcp and ck can be issued from any UNIX machine, as well as on the VMS server. The "mcp" command is just passed directly to the server machine. However, there are a number of "ck"-type commands that are also performed on the UNIX side, and these are tried first, then the ck command is passed to the VMS server if the UNIX command file does not recognize the input parameter.

UNIX Commands

The UNIX command files are now located a /usr/dsp/com2 directory (just as the executables are in /usr/dsp/bin2). The pathgac* , *ingest* and rcp_control.sh files are used by the processing, not interactively. The ap.defs files just set up the environmental variables for the processing. So, that leaves just three files:

exa - runs the full examine on an image (to retrieved full header)

start - starts the processing on various machines

ck - performs various kinds of checks

The strt command file should take two parameters: the type or types ofmcps to start, and the computer to start them on. The remote start is not quite working. The type(s) of mcps to start are:

u1	s1	o	I	d	
w1	w2	w3	w4	fw	ew
u2	u3	u4	s2		
us	uso	ii	iii		
3uso	4uso	4u2so	usoii	usoiii	

The ck command file checks for a number of options on the UNIX side. If the command is not recognized as one it knows, it passes the whole command line to the VMS side.

Processing Overview

The input dataset consists of a stream of datafiles representing orbits of a satellite. The satellite circles the earth in about 100 minutes. As it circles, the earth turns beneath it, so that the edges of succeeding orbit overlap slightly. In addition, each orbit is recorded

by one of two data recorders; one is started (ideally) shortly before the other finishes, so that the end of one orbit overlaps slightly with the beginning of the next orbit.

After about 14 orbits, the earth has rotated once (ie, one day has passed). This means that the satellite has taken data over the entire earth TWICE, once on an ascending (south to north) leg and once on a descending (north to south) leg. For the NOAA polar orbiters, the ascending leg corresponds to daytime and the descending to night.

On the archive device, these data files are stored in increments of a yearday (the first five digits of the filename), and the various jobs that store, retrieve and access the data are usually organized in the manner. To date, this has been the most convenient arrangement, with one day's worth of data fitting comfortably on the staging disks while still "occupying" the processing streams.

The processing scheme can be though of as addressing a number of tasks, which can be sorted into characteristic elements defined by the input data. In the AVHRR GAC processing, the tasks and their data elements are:

Task	Increment	Components
1. Spooling data	yearday	14 orbits
2. Autoproc entry	orbit	self-contained
3. Ingest/atmospheric corr/	orbit	6 pieces of orbit
4. Spacebin	orbit	6 pieces "
5. Orbit	orbit	6 pieces "
6. Daily	3 yeardays	14*3 orbits
7. Weekly1	1 week	7 daily files
8. Weekly2	3 weeks	3 weekly files
9. Weekly3	1 week	weekly reference file
10. Weekly4	1 week	cloud-masked files

1. Spooling data: This job acts as a daemon, checking to see if files are needed, and that disk space is available. If so, it copies one day's work of files from one device to another. Most of the spooling jobs are submitted once and recycle in increments of one day.

2. Autoproc entry: This job notifies the db that a data file exists and is ready for processing. It deals with a single file at a time.

3,4, Ingest/atmospheric correction and spacebin: These jobs work on individual pieces from a single orbit file. They can be thought of as a set of tasks that must be performed in sequence to a single piece. The tasks are separated to enable a degree of overlapped processing, that is, one piece is being ingested while the last piece that was ingested is begin atmospherically corrected, and the piece just corrected is being spacebined. (See short section later for explanation of each of these steps.)

5. Orbit: When all pieces of an orbit file have been finished (through spacebinning) a job is run that gathers the pieces into single files for transfer to a remote disk. This consolidation produces (usually) one ascending and one descending file for the given yearday, plus POSSIBLY a third file with data to be included in either the previous or next day. (Omitting the long explanation, the way a "data-day" is defined may result with data from the same piece of a pass being put into different "days".

6. Daily: : When all orbits or segments of orbits due to the data-day definition of a given yearday file have completed processing (through orbit), a job is run that gathers the pieces into single files for transfer to a remote disk. This consolidation produces (usually) one ascending and one descending file for the given yearday

7. Weekly1: When all seven days of a give week have been created, the ascending and descending data are combined into weekly asc and dsc files.

8. Weekly2: When the weekly files for three weeks in succession have been created, these are combined into 3-week asc and dsc files, and these files are run through a gap-fill program to eliminate cloudy areas. These are called the asc and dsc reference files for the middle week of that three-week period.

9. Weekly3 & 4: After the reference files for a given week have been produced, these are used to cloud-mask each daily file for that week (asc ref used on asc dailies, dsc on dsc dailies). The cloud-masked daily files are then recombined into cloud-masked weekly files, and a large number of product files are extracted from the daily and weekly declouded files.

Ingest/AtCor/Spacebin: Ingest refers to the process of extracting a part of an input orbit and storing it in a dsp-format file, including the navigation from the orbit file itself. After ingest, a series of procedures is run on the ingested file to apply various corrections to the navigation (this is called the sector process, which is actually a series of processes). No new file is produced in the sector process. The atmospheric correction part takes the input data (radiance in five different spectral bands) and produces an SST estimate at each point in the input ingest file. These files are still at the 4 km resolution, and are in what is called "satellite perspective", ie, the successive scan lines. The spacebin job takes the input satellite perspective data and bins it into a global 9 km equal area grid. Note that these files contain entries only for bins with data. Successive jobs combine or treat the data using these bins.

Simplifications in the schema are planned, particularly in the recipes (use a single table) and main/pc records (elimination of extraneous fields).

There are three sets of table in the processing database. One set contains three tables that store information on the processing, defining the steps and what is to be done in each of the steps. Another set has two tables that track each of the processing entities and the jobs associated with them, and a third set contains various tables that provide ancillary information for definition and validation purposes.

These types contain these tables:

PROCESSING RECIPE TABLES:

RECIPES
PROCESS_STEPS
PARAMETERS

JOB ENTRY TABLES:

MAIN
PROCESS_CONTROL

MISC. DB TABLES

BOOK_KEEP
COMPUTERS
DAYINFO
SATELLITES
SATSEN
SENSORS
USER_GROUP
WEEKINFO

Obsolete tables: ARCHIVE, ARCH_INFO, CHANNEL, FORMAT, INGEST_ERROR, LASER_FILE, LOCALDIRS, MEDIUM, PC_COMPLETE, PROCESS_ERROR, PROJECT, RAW_DATA, SOURCE, STATION, TRANSMISSION

PROCESSING RECIPE TABLES - RECIPES/PROCESS_STEPS/PARAMETERS:

There are three tables that are used to define the processing stream: the RECIPES table, the PROCESS_STEPS table and the PARAMETERS table. (Future implementations will combine these into either one or two tables.) The RECIPES table contains an entry for each integral step in the processing stream. That is to say, a RECIPE defines a step or set of steps that are to be performed as a unit. The PROCESS_STEPS table contains a set of entries for each RECIPE, defining the processing steps, and the order in which they are to be performed. The PARAMETERS table provides additional definitions that may be used in the PROCESS_STEPS table.

RECIPES Table:

A processing RECIPE may contain more than one processing step. The steps defined by a RECIPE are to be performed as a unit, without interruption, and the results of the processing transmitted to the processing db at the end. The fields of the RECIPE table are used to assign characteristics and control some aspects of the job triggering. The fields are:

Table RECIPES	
recipe_code	Code number for keyword retrieval
recipe	Recipe name

def_priority	Default priority to assign to procedure
process_class	All jobs with the same "svctype"
computer_class	When one job is assigned to a computer, all related jobs are as well
trigger_class	Trigger_class of this recipe
class_to_trigger	Trigger_class to trigger when present work is completed

Values for these fields are:

CD	RECIPE	PRI	PROCESS_CLS	COMPUTER_CLS	TRIGGER_CLS	CLS_TO_TI
1	'GAC_INIT	1	'INIT'	'INGATSB'	'INIT'	'INGATCOR
2	'GAC_INGATCOR'	1	'UNLIMITED'	'INGATSB'	'INGATCOR'	'NONE'
3	'GAC_SPACEBIN'	1	'SBIN'	'INGATSB'	'SINGLEREC'	'TIMEBIN
4	'GAC_ORBIT'	1	'ORBIT'	'INGATSB'	'TIMEBIN'	'DAILY'
5	'GAC_DAILY'	1	'DAILY'	'WEEKLY'	'DAILY'	'NONE'
6	'GAC_WEEKLY1'	1	'WEEKLY1'	'WEEKLY'	'WEEKLY1'	'WEEKLY2

The process_class field refers to the job type. That is, when mcp requests a job, it requests a job with a particular process_class. This was originally intended to be used if one mcp would be allowed to run different types of jobs, but has operationally become limited to a single type of job for each mcp. In the example, there are six different process_classes representing five mcp types: mcp-i, mcp-u1, mcp-s1, mcp-o, mcp-d and mcp-w1.

The computer_class is used to control which computer performs the processing. For example, the processing of a single orbit should all occur on the same computer, as the input file must be copied to a local disk, and this should be done only once. So, when an orbit is assigned to a computer, all tasks with the "INGATSB" computer class will only be run on that computer. In the example, when the INIT job is assigned to a particular computer, all other jobs with the same computer_class (INGATCOR, SPACEBIN and ORBIT) are also assigned to that computer.

The trigger_class and class_to_trigger fields are used in the triggering of follow-on jobs. When a job completes successfully, the process_status for that collection of jobs (ie, all process_control records with the main_link) is checked. If there are no submitted or aborted jobs, then the "class_to_trigger" field is checked for the recipe just completed. If it is "NONE" no triggering of this type is

performed; otherwise, all records with the specified trigger_class are submitted for processing. In the example, when the INIT job completes, all INGATCOR jobs for that orbit will be submitted for processing.

PROCESS_STEPS Table

This table set the information retrieval, workspace definition, and command files to be run, and their order, for each recipe to be used in a processing stream. The fields of the PROCESS_STEPS table are:

```
!  RECIPE         Steps are assigned to this procedure
!  process_step   Process step number
!  command        Command to execute
!  jump           Jump to this step if appropriate
```

Each recipe will have a number of steps defined for it.

RECIPE	PROCESS_STEP	COMMAND	JUMP
'GAC_INIT'	1	'NAME_MAKE'	0
'GAC_INIT',	2	'pathgac_init',	0
'GAC_INGATCOR'	1	'NAME_MAKE'	0
'GAC_INGATCOR'	2	_WEEKINFO'	0
'GAC_INGATCOR'	3	'SET_INGEST	0
'GAC_INGATCOR'	4	'WSO:nlmc'	0
'GAC_INGATCOR'	5	'pathgac_ingator'	0

There are three types of things that can be defined in a step.

1: Certain commands (such at SET_INGEST or GET_ASCDSC) direct the server to retrieve information from the database, and write out workspace variables.
(Examples: Steps 1 in GAC_INIT and 1-3 in GAC_INGATCOR above.)

2: A command may be simply the name of a dsp command file to execute.
(Examples: Steps 2 in GAC_INIT and 5 in GAC_INGATCOR above.)

3: A step can be used in conjunction with the PARAMETERS TABLE to assign a particular value to a workspace variable. (Example: Step 4 in GAC_INGATCOR above - see also PARAMETERS below.)

PARAMETERS Table

The PARAMETERS table contains three fields:

command	command for which variable is defined
work_space	workspace variable to define
data_value	value to assign to variable

and sample values are:

COMMAND	WORK_SPACE	DATA_VALUE
'WSO:nlc	'SSTTYPE'	'nlc'
'WSO:nlsst'	'SSTTYPE	'nlsst'
'WSO:nlmc'	'SSTTYPE'	'nlmc'

In the PROCESS_STEPS example, step 4 in GAC_INGATCOR is requesting that a workspace variable, 'SSTTYPE', be assigned a value of 'nlmc' before the dsp command procedure 'pathgac_ingatcor.dsp' is run.

JOB ENTRY TABLES:

MAIN Table

This table tracks the "entities" that require processing. There is one entry for each "class" of jobs that need to be run. For example, there is one record for each satellite orbit to be processed, and the file_name is the input orbit file name. There is another entry for all jobs that need to be run on a daily basis (ie, jobs that produce files in increments of one day), and another for weekly jobs, etc. The file names of these are yyddd, yyww, etc.,

There will be at least one record in the PROCESS_CONTROL table for each record in the MAIN table. If multiple jobs are to be run on the same "entity", there will be one PROCESS_CONTROL record for each job.

Field	Comment
record	Main record number
file_name	Link to several relations
file_number	File number on archive medium
raw_data_link	Link to RAW_DATA table
archive	Validate with ARCHIVE table
archive_label	Link to ARCH_INFO table
format	Validate with FORMAT table
satsen_code	sat/sen code number
satellite	Type of satellite - validate with SATELLITE table
sensor	Type of sensor - validate with SENSOR table
transmission	Transmission - validate with TRANSMISSION table
channel	Channels in input file - validate with CHANNEL table
orbit	Satellite orbit number
pass_time	Timestamp at beginning of pass
pass_end	Time of last scan line in the pass
yearday	Yearday of file
scans	Number of scan lines in scene
miss_scan	Number of missing scan lines in scene

source	Data source - validate with SOURCE table
project	Project - validate with PROJECT table
release_link	Main record to notify upon completion
release_recs	No. of release_recs this is waiting for
release_done	No. of release_recs done
process_recs	No. of process_control recs for this entity
process_done	No. of process_control records completed
map	Bitmap of 10 X 10 degree coverage
QC_status	Status of quality control
who_code	Who modified; Validate with USER_GROUP table
last_mod	Date of last record modification
audit	Date of record creation

PROCESS_CONTROL Table

This table stores information on individual jobs through the automatic processing system.

Field	Comment
record	Internal record number of file to process
main_link	Link to MAIN relation
recipe	RECIPE to be executed Validate with RECIPE table
process_step	Last process_step completed (obsolete)
satsen_code	
satellite	Validate with SATELLITE table
sensor	Validate with SENSOR table
computer	Computer assigned to process this PCR; Validate with COMPUTER table
computer_class	Used to restrict jobs to one computer
process_status	Is the job in HOLD, SUBmitted, FINished or DSPAbort status
process_class	MCP-type class of this job
rec_to_trigger	PC Record to trigger when this completes if zero, this is triggered by another if -1, neither triggers nor is triggered
trigger_class	The trigger_class of this job
class_to_trigger	The class to be triggered when this class has completed

priority	Priority to assign to job
totscan	Total scans in WHOLE SCENE
begscan	Beginning scan line of this piece
endscan	Ending scan line of this piece
yearday	Yearday from MAIN_LINK's input file
miscinfo	Miscellaneous information (varies)
source_file	Output file for this PCR.
source_directory	Directory of input file.
who_code	Validate with USER_GROUP table
last_mod	last record modification
audit	record creation

MISC. DB TABLES

BOOK_KEEP Table

The BOOK_KEEP table is a special-purpose table used to keep track of the current number of entries in those tables that are permitted to "grow. The three fields in the BOOK_KEEP table are:

record	record number in this table
relation_name	name of relation
max_record	current number of records

and INITIAL entries are:

RECORD	RELATION_NAME	MAX_RECORD
1	'BOOK_KEEP	4
2	'RAW_DATA'	0
3	'MAIN'	0
4	'PROCESS_CONTROL'	0

The BOOK_KEEP entry has an initial value of 4 because four tables are currently used in this manner - the three "data" tables and BOOK_KEEP itself.

When a new entry is added to any of these tables, the BOOK_KEEP value for MAX_RECORD for that table is first incremented, then that value assigned as the record number for that table. This is the APServer's method of assigning distinct record numbers to these tables.

COMPUTERS Table

The COMPUTERS table is used to validate computer names using the APServer system, and to provide information about them. The fields in the COMPUTERS TABLE are:

Field	Comment
computer	Valid keyword value
abbrv	Abbreviation
os	Operating system
site	Location of computer
*The following fields are not currently being used.	
run_status	Running status of computer (run, pause, finish, stop)
alarm_status	Whether the start/stop alarm is to be used (n/y)
alarm_start	When to begin processing (hhmm) (LOCAL time)
alarm_mcps	Which mcps to start (list of start files, sep by comma)
alarm_stop	When to finish up processing (hhmm) (LOCAL time)

and typical entries are:

COMPUTER	ABBRV	OS	SITE
MARIAH	mar	VMS	RSMAS
andrew	and	UNIX	RSMAS
enuka	enu	UNIX	RSMAS

There are also a number of fields that have not yet been implemented, that will be used to automatically start and stop processing on a given machine.

DAYINFO Table

This table is used to define the start the ascending and descending data-days for each satellite. This table also assigns a week to each day (which can be defined separately by satellite). The fields are:

Field	Comment
yearday	yearday
satsen_code	sat/sen code number
week	Week to which this day is assigned
asc_beg	Beginning of day for ascending data
dsc_beg	Beginning of day for descending data

and typical entries are:

YRDAY	SATSEN_CODE	WEEK	ASC_BEG	DSC_BEG
88308	1	8845	88308011124	88307135119
88309	1	8845	88309010320	88308134417
88310	1	8845	88310021038	88309133721
88311	1	8845	88311020251	88310133021
88312	1	8845	88312015525	88311132309
88313	1	8845	88313014817	88312131539
88314	1	8845	88314014120	88313130747
88315	1	8846	88315013424	88314141507
88316	1	8846	88316012720	88315140707
88317	1	8846	88317012000	88316135929
88318	1	8846	88318011220	88317135211
88319	1	8846	88319010417	88318134508
88320	1	8846	88320021135	88319133811
88321	1	8846	88321020345	88320133112

YRDAY	SATSEN_CODE	WEEK	ASC_BEG	DSC_BEG
88308	2	8845	88308044643	88307...
88309	2	8845	88309043751	88308...
88310	2	8845	88310042925	88309...
88311	2	8845	88311042123	88310...
88312	2	8845	88312041338	88311...
88313	2	8845	88313040558	88312...
88314	2	8845	88314035812	88313...
88315	2	8846	88315035008	88314...
88316	2	8846	88316034139	88315...
88317	2	8846	88317044825	88316...
88318	2	8846	88318043931	88317...
88319	2	8846	88319043103	88318...
88320	2	8846	88320042300	88319...
88321	2	8846	88321041514	88320...

Entries need only be made for a particular satellite for the life of that satellite, not for all days defined in the table.

SATELLITES Table

This table is used to validate satellite names and store associated data. The fields are:

Field	Comment
satellite	satellite name
extension	file extension for ingested files
sat_id	NORAD satellite number
prefix	prefix for processed files

and typical entries are:

SATELLITE	EXTENSION	SAT_ID	PREFIX
'NOAA-9'	,'NO9'	15427	'K'
'NOAA-10'	'N10'	16969	'J'

The SATELLITES table is linked to the SENSORS table by the SATSEN_CODE table.

SENSORS Table

This table is used to validate satellite sensors.

Field	Comment
sensor	Valid keyword value

SENSOR				
'AVHRR'	'HIRS2	'MSU'	'SR	'VHRR'
CZCS	'DCS'	'ALT'	'HCMR'	'MSS'
'OLS'	'SAR'	'SCAT	'SMMR'	'TM'
'VAS'	'VIR'	'VISSR'	'SEAWIFS'	

SATSEN Table

This table is used to validate satellite/sensor combinations, and assign a specific code to each pair. It is used to link the SATELLITES and SENSORS tables.

Field	Comment
satsen_code	sat/sen code number
satellite	satellite name
sensor	sensor name

SATSEN _CODE	SATELLITE	SENSOR

1	'	NOAA-11'	'AVHRR'
2	'	NOAA-9'	'AVHRR'

USER_GROUP Table

This table is used to validate members of the user USER_GROUP.

Field	Comment
code	Code number for keyword retrieval
user_name	Computer user_name ID
full_name	Users' full name
telephone	Phone
site	Where the user is
email_address	E-mail address
telemail_address	TELEMAIL address

Note: The autoprocessing interface automatically loads the USER_NAME into this table when a new user first runs the system. All other information (full name, site, etc.,) must be loaded separately.

Code	User_Name	Full_Name	Telephone	Site
0.	UNDEFINED			
1.	VICKI	VICKI M. HALLIWELL	305-361-4178	RSMAS

Code	User_Name	email_address	telemail_address
1.	VICKI	vicki@miami.rsmas.miami.edu	NONE

WEEKINFO Table

This table is used to define the 4-week and 8-week time periods, and the directory for the daily files for each week.

Field	Comment
week	The week
fourwk	The 4-week interval for this week
eightwk	The 8-week interval for this week
wkdir	Directory for the daily files by week

WEEK	FOUR WK	EIGHT WK	WKDIR	WEEK	FOUR WK	EIGHT WK	WKDIR
0	104	108	'/(disk)/wk/wk01'	27	2528	2532	'/(disk)/wk/wk27'
1	104	108	'/(disk)/wk/wk01'	28	2528	2532	'/(disk)/wk/wk28'
2	104	108	'/(disk)/wk/wk02'	29	2932	2532	'/(disk)/wk/wk29'
3	104	108	'/(disk)/wk/wk03'	30	2932	2532	'/(disk)/wk/wk30'
...(entries omitted)							
24	2124	1724	'/(disk)/wk/wk24'	51	4952	4952	'/(disk)/wk/wk51'
25	2528	2532	'/(disk)/wk/wk25'	52	4952	4952	'/(disk)/wk/wk52'
26	2528	2532	'/(disk)/wk/wk26'	53	4952	4952	'/(disk)/wk/wk52'

The following tables are currently included in the processing database, but provide no function, and will be eliminated or moved to other dbs in the future.

ARCHIVE
ARCH_INFO
CHANNEL
FORMAT
INGEST_ERROR
LASER_FILE
localdirs
MEDIUM
PC_COMPLETE
PROCESS_ERROR
PROJECT
RAW_DATA
SOURCE
STATION
TRANSMISSION
TRIGGERING.INFO`----------|

The triggering is in the db_report subroutine.

1. When the end of a job is reported to the database, the value of "status" and the existence of an error message is checked. If either of these indicate an error, then the process_status for that process_control record is set to 'dspa', and no triggering occurs. Otherwise, process_status is set to 'fin', and the triggering is tested.

2. If the rec_to_trigger for this process_control record is greater than zero, it is marked 'sub' (submitted for processing).

Example - Completion of Ingatcor triggers Spacebin:

record	recipe	file	rec_to_trigger
1001	Ingatcor	90123123456.data	1002
1002	Spacebin	90123123456.data	-1

The completion of record 1001 will trigger the submission of record 1002.

3. If the rec_to_trigger is less than 1, the "class_to_trigger" for that record is checked. If it is "none", no further class triggering is done. Otherwise, the process_control records for that main record are checked. If any records are marked 'sub' or 'exe, no triggering occurs. Otherwise, all process_control records associated with that main record with the matching trigger_class are submitted for processing.

Example - Completion of Init triggers multiple Ingatcors:

record	recipe	trigger_class	class_to_trigger
1000	Init	Init	Ingatcor
1001	Ingatcor	Inhgatcor	None
1003	Ingatcor	Inhgatcor	None
1005	Ingatcor	Inhgatcor	None
1007	Ingatcor	Inhgatcor	None
1009	Ingatcor	Inhgatcor	None
1011	Ingatcor	Inhgatcor	None

When the Init job complete, all six of the Ingatcor jobs will be submitted.

Three special cases:

1. If the process_class is "orbit", the orbits for that yearday are checked. If any are unfinished, no triggering occurs. If all are finished, the previous and succeeding days are checked. If all are finished, then the rec_to_trigger of the middle day is submitted. The previous and succeeding three-day periods are also checked if needed.

2. If the process_class is "daily", all the daily jobs for that week are checked. If any are unfinished, no triggering occurs. If all seven days for that week are done, then the first weekly job (WEEKLY1) is submitted.

3. If the process_class is "weekly2", the "weekly1" jobs for that week, the previous week and the succeeding week are checked. If any are unfinished, no triggering occurs. If all are finished, then the rec_to_trigger for the middle week is submitted. The previous and succeeding 3-week periods are also checked.

B.2 Client/Server Status (S)

The following functionality was added to the client/server environment during this quarter.

1. The ap program was used to process AVHRR data. About 80 days of data was processed using the modified program. This was done when weekly products were first automated to be processed by ap. Prior to this modification, all the weekly products had been done manually

2. A data processor, similar to ap but simpler was written. As part of this effort, the frame for shell scripts is finished. ap is complicated, especially the client-server communication mechanism. All the communication depends heavily on the robustness of the network software. This makes ap somewhat vulnerable in a sense that its robustness is measured by the weakest link in the network software and its own traffic handling ability. So a simpler system would be more desirable, one that can be easily maintained and easily adapt to different data processing jobs.

3. A scan program was written which scans Tiros data files and gathers the necessary information for later use as part of the automatic processing. This program scans the incoming data file and prepares the input control files for ingester. This is an important step in the AVHRR data processing. In ap, it is done on the VMS side while all the other processing activities are on UNIX side. A UNIX version of scan program could shorten the processing path and make the whole process more efficient.

4. The software interface for ap and DLT tape library was rewritten. The use of new coefficients to process AVHRR data and to accommodate a new, unforeseen addition caused the interface between ap and DLT tape library to be modified.

5. An effort is underway to write an orbital data processor; the processor is partially tested. Bob instructed that a new data processing system should be written with Warner providing design guidance. As a result of the design meetings, an independent orbit data processor is being written that could make ap more efficient.

B.3 Matchup Database (P)

B.3.1 1st Quarter Matchup Activities

During this period, AVHRR in-situ matchups were completed for the period 1985–1993. This covers the NOAA-9 period completely (1985–November 1988). In preparation for public release of the matchups through the PO_DAAC at JPL, documentation was prepared describing the compilation and composition of the distributed databases. The matchups will be transferred to JPL in the immediate future.

Other activities involved the compilation of additional in situ data to extend the matchups. Drifting buoy data were acquired from Canada's Marine Environmental Data Service (MEDS) for the period January–October 1994 (the rest of 1994 is not available yet). An entirely new set of buoy data was purchased from the United Kingdom's Meteorological Office. Data for these buoys encompass the period 1991–1994. Although the 1991-93 data were not included in the current version of the matchups, they may be used for independent algorithm validation.

Finally, the matchup databases were used to test a new approach to the estimation of SST algorithm coefficients. This approach, currently being reviewed by the SST Science Working Group, involves the estimation of coefficients on a month-by-month basis, using a 5-month set of matchups centered at the month in question. This approach helps to lower the biases and increase in variability associated with SST estimates after the explosion of Mount Pinatubo (June 1991). Furthermore, low-frequency temporal trends such as those detected for the NOAA-9 data set also seem to be reduced by this approach.

B.4 DSP Support (M)

B.4.1 Testing:
None listed

B.4.2 Modifications/Additions to DSP:

SEAWIFS:
ANLY8D: Change from 'cal.hdf' to new distribution file name 'SEAWIFS_CAL.TBL'.
Don't transpose arrays from get_l1a_record. GSFC changed routine to match spec.
Version 2 of Gordon/Wang atmospheric algorithm. Minor corrections to version 2 atmospheric algorithm.
Add O_2 correction for SeaWIFS band 7.
Corrections/space savings to version 2 atmospheric algorithms.
Add new Gordon tau routine.
Collapse SeaWIFS library dependencies into one symbol.
Output flags_pc as 'percent input pixels' instead of 'pixels'.
Convert La to La(single scattering) for tau calculations.
Correct phi table lookup, must use abs(phi) to locate bounding entries.
Document error in existing phi tables (missing 0-4.99 deg data).
Optimize repetitive calculations based on 'ss11'.
Correct for alpha/sgi builds. Add test program for gordon_tau_a.f.
Add diagnostic printing statements for OSF debugging.
Fix formation of aerosol/phase function file names.
Limit calculation of values that could overflow.
Add test programs for wang2 and gordon_tau routines.
Test programs for wang2 and gordon_tau routines.
Rename variables to clarify usage and add diagnostic print statements.

Convert to v4.2 library calling structure/ conventions.
Track interface changes to l2w routine.
Output radiances instead of counts for "bad" pixels.
Disable NaN debugging code.
Remove outdated routine (l2_calib.rat).

BIN Programs:
SSBIN-HDF: Use tilt; both gsfc and miami quality determination; use gsfc v4.1 i/o routines, don't use land mask (done in Anly).
STBIN-HDF: Use tilt; both gsfc and miami quality determination; changes for gsfc v4.1 i/o routines.
SMAP9-HDF: Use tilt; both gsfc and miami quality determination; changes for gsfc v4.1 i/o routines. Add parameter file; use v4.2 SeaWiFS I/O routines.

B.4.3 Problems fixed:
QUORUM: Add curlin to a debug. Fix to work on VMS. Strengthen missing line checks. Comment out diagnostic print statements.
INGEST/LIB/GETPSEUDOTIME.C: Add EOS to message.
INGEST/LIB/SCANFORSYNCH.RAT: Add debug message (ifdef'd).
INGEST/LIB/READRECORD.RAT, SCANBACK.RAT: Look for next synch pattern using scanback.
INGEST/LIB/FFCSWITCH.RAT, READRECORD.RAT: Fix to work on VMS.
LOADDB: Declare variable used on VMS.
QRMPACK: Fix to work on VMS. Write symbol PACK$FILENAME with output file name.
MIA2GIF: Re-initialize pointer used by GetPixel every invokation. Re-initialize more variables. Add decimate option.
LIB/IO/VAX_EXTRACT.C: Handle VAX -> IBM PC convertion.
PATHNLC: Use a coefficient file for all satellite/date combinations. Make sure to read whole comment in coeffs file. Add new satellite zenith angle and temperature test and use bit one in mask1.
SHPSPH: Fix change of load commands to use input x,y instead of y,x. Modify use of variables so "x" is element and "y" is scan.
DBMAN: Put EOS after *.book and *.rec file names.
PATHBIN: Add debugging code to check for NaN's (Not-a-Number). Disable NaN debugging code. Skip bin calculation loop if pixel is not supposed to be binned; fix some of the indenting. First part of left edge/right edge binning changes. Use new SATZTEMP flag as part of "satellite" test.
SCRIPP: Strengthen missing line algorithm. Comment out diagnostic print statements.

MACE2: Write out image start time instead of current time in dim file data lines.

B.5 Direct Project Support

B.5.1 SeaWIFS **(S)**

B.5.1.1 1st Quarter Efforts

Received new ancillary data routines. Jim integrated into ANLY, discovered problems with HDF routine when trying to access all 4 data fields, routine returned zero data. SeaWiFS resolved problems and new routines successfully integrated.

Received new L1 and L2 HDF routines. Working with SeaWiFS project to resolve difficulties in linking routines with ANLY.

3/23 Carder Telecon algorithm code to be available by end of march - Steve (SKA)

3/17 Jim able to link test program with HDF routines, sue to continue working with test program to verify L2 read and write with typical ANLY files.

Jim is restructuring ANLY to separate program sections dealing with calibration, navigation, I/O, atmospheric correction, and product generation. This will allow program to use either CZCS or SeaWiFS I/O and data.

Jim computed Rayleigh tables for all SeaWiFS bands, added ozone tau coefficients, Rayleigh tau coefficients, and computed data day limits for SeaWiFS project.

B.5.1.2 **Outstanding items:**

B.5.2 MODIS **(M)**

Modis Ocean integration: will split efforts between SDST and Miami where I/O and toolkit functions will be integrated by SDST. Miami will deliver Beta code using SeaWiFS modules and related I/O.

Test data sets will use SeaWiFS simulated data produced by W. Gregg. Real SeaWiFS data will be substituted when available.

MODIS algorithms are represented by SeaWiFS coding. MOCEAN team member will need to supply actual MODIS modules during the next quarter to prepare for re-coding to EOS standards. SeaWiFS code presently in RATFOR. MODIS algorithms will be coded in Fortran-90. A beta compiler is available from DEC for the DEC/ALPHA machines. Discussions are underway to acquire a similar capability for the SGI machines.

Discussions in progress with SWAMP for gridding standards and data day definition. Present EOS standard is to use ISSCP equal area defined at 1.25 degree and then use equal angle sub-division for higher resolution; data day definition uses 24 hour. The approach defined by the MOCEAN team, e.g. use ISSCP for all resolutions and use of spatial data day definition is being proposed as an alternative.

C.5.2 Continue timing tests with CZCS and SeaWiFS algorithms.